5A-01

進化を考慮したデータ基盤の設計および自動構築

宿谷　琴子†　　土肥　実久†　　福山　訓行†
株式会社富士通研究所　スーパーミドルウエア・ユニット†

1.　はじめに

日々変化していく市場に素早く対応するため、企業が保有するデータからの価値抽出、すなわち「データ利活用」が求められている。例えば、ある駅でのカメラの映像データから人間の行動分析を行い、時間帯や気象条件によってどんな人が利用するのかを出力し、その結果によって駅構内の小売店の商品の陳列を変える、ということも可能になる。

現状、データ利活用の各処理の多くは手作業で行われているが、データドリブンなビジネスを素早く回すためには処理の自動化が必要である。データ利活用を本番環境で繰り返し自動実行する場合には分析用プログラムだけではなく、データの処理を実行できる環境を含めたシステム上で、目的に応じた様々な分析手法を模索することになる。しかし、分析手法を取り替えた際のシステムの再構成、制御情報の更新は手間がかかる上にインフラ構築の専門知識を要するという課題がある。

本稿では、分析手法を模索する際に、インフラ構築の専門知識を必要としないためのデータ基盤の設計について述べる。

2.　データ利活用システムの課題分析

データ利活用を行うためのインフラからアプリの全体を含むシステムを本稿では「データ利活用システム」と呼ぶ。分析手法を模索するためのマイクロサービスを意識した先端のデータ利活用システムは、図１に示すように、物理サーバやコンテナ基盤などの「インフラ」の上に、データソースからの取込み、データ加工、データ分析という処理を実行するデータフロー、処理に用いるアプリ、データフロー上の処理を順番に実行させるためのデータフロー制御、使用するデータを蓄積するためのデータストアから構成される「アプリシステム」が乗っている構成が増えてきている。

このシステム上で、分析手法を取り替えるためにデータ分析アプリなどを変更する際にはデータの処理方法が変化するため、データフローを変更する可能性がある。このとき、変更前と同じデータを使用したい場合はアプリシステム全体を新規に作り替えるか、変更部分をシステムに合わせて個別に修正するため手間がかかってしまうという課題がある。

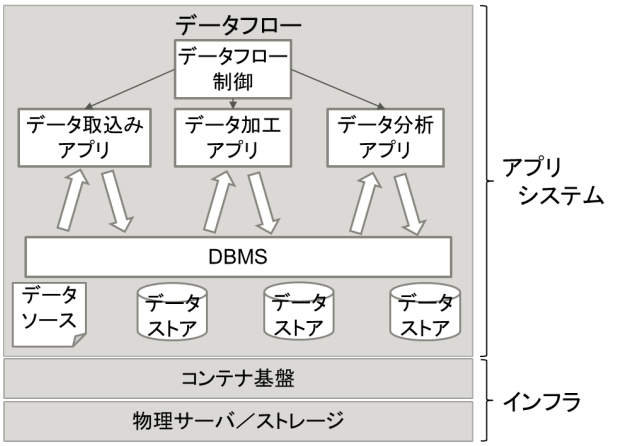

図１：データ利活用システムの例

この課題に対する解決策として、アプリシステムのレイヤーを「作り直す部分」と「作り直さない部分」に分ける。

具体例として、図２に示すユーザーがデータ分析アプリ A を別のデータ分析アプリ B に変更した場面を想定する。このとき、分析に用いるデータを蓄積したデータストア＝ストレージのパスワードはランダム発生する場合もあり、単純にデータ分析アプリを取り替えようとしても同じデータストアに接続できずスムーズな再起動ができなくなってしまう。

このケースにおいて、ユーザーが直接変更を行う、つまり「作り直す部分」はデータ加工アプリとデータフローである。また、ユーザーが直接操作する必要が無い、すなわち、「作り直さない部分」であるデータフロー制御部分からインフラ部分までの間のレイヤーは、本来、意識しなくても良い部分である。本稿ではこのレイヤーを「データ基盤」とすることで、ユーザーにこの部分を意識させないようにできる可能性があることに着目した。

これを実現するには、データ基盤は、まずユ

ーザーがインフラ部分に作成したストレージの設定などを保管する必要がある。次に、ユーザーがデータ分析アプリを A から B へ変更した後、変更前と同じデータを用いた新しい分析を行いたいとき、保管しておいた設定を用いてデータストアとデータ分析アプリ B との接続を構築しなければならない。これにより、ユーザーは同じストレージからデータを使用できる。

このように、「作り直さない部分」をデータ基盤に吸収することによって、ユーザーがデータフローやアプリの変更を行うだけでデータ基盤はその変更を適用し、変更前と同じデータを用いて新しいデータフローを自動で実行できるようになる。

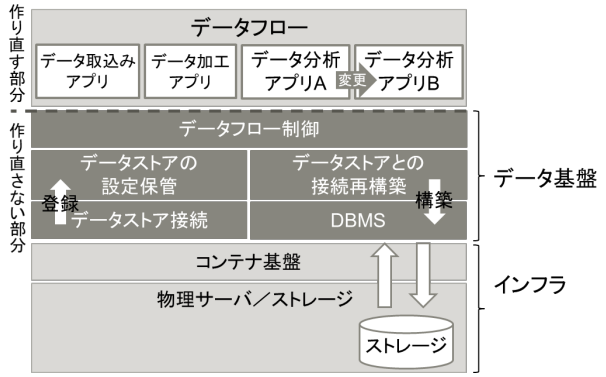

図２：提案するデータ利活用システム構成

3. データ基盤の要件定義

前述のデータ基盤は、作り直さない部分を吸収することで逆に作り直す部分を簡単に何度も作り直せることを目的としている。このデータ基盤を実現するための要件は以下のように定義される。

【必須要件】

・データフローやアプリが新しく作り替えられた際、その変更を適用できる

・永続的なデータの蓄積ができる

【選択的要件】

データソースからの取込み、データ加工、データ分析という処理が確実に自動実行できる

4. データ基盤の設計

前述の要件を満たすため、図３に示す構成を実装した。ここで、DBMS として Postgres を、コンテナ基盤として Kubernetes を、データの処理を行うアプリは Docker コンテナで実装する方式を採用した。

【必須要件】に対して、「記録部」と「再現部」を導入する。「記録部」は初回時にユーザーがデータ基盤にデータストアを設定したときのホスト名、ユーザー名、パスワード、データベース名などのパラメータを「記録部」に配置した RDB に記録する。また、「再現部」はデータフローやアプリが新しく作り替えられた際、「記録部」に記録したパラメータを Kubernetes に渡して DBMS を起動する。この 2 つの機構により、ストレージの認証情報を維持することが可能になるため、アプリが変更されても同じデータを扱うことができる。

上記の【必須要件】に加えて【選択的要件】では、ワークフロー管理システムと記録部／再現部を連携させてデータフローの制御を行うことで、データフローの維持が可能になる。

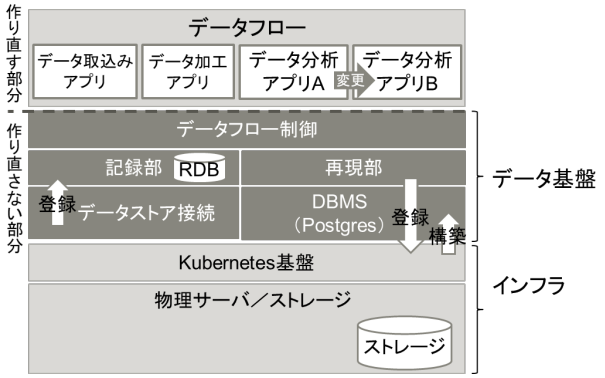

図３：データ基盤の実装例

5. まとめ・今後の予定

本稿では、インフラ構築の専門知識を必要とせずに最適な分析手法を模索するため、アプリシステムをユーザーが作り直す部分と作り直さない部分とで分けることで分析手法を簡単に取り換えられるデータ基盤の設計を行った。

この設計に従いデータ利活用システムを構築し、アプリを変更しても変更前と同じデータが使用できることを検証する。

また、データ基盤のレイヤーは増える可能性があるため、実際にデータ利活用システムを用いてデータ分析や機械学習を行なっていく中でデータ基盤に必要な機能を追加し改善していく予定である。

将来は、データストアを変更する場合においても永続的にデータを蓄積できる設計を目指す。